


```

LL               IIIIII               SSSSSSSS
LL               IIIIII               SSSSSSSS
LL               II                  SS
LL               II                  SS
LL               II                  SS
LL               II                  SS
LL               II                  SSSSSS
LL               II                  SSSSSS
LL               II                  SS
LL               II                  SS
LL               II                  SS
LL               II                  SS
LLLLLLLLLLLLLL  IIIIII               SSSSSSSS
LLLLLLLLLLLLLL  IIIIII               SSSSSSSS

```



```
1 0001 0 MODULE BINDVL (
2 0002 0 LANGUAGE (BLISS32),
3 0003 0 IDENT = 'V04-000',
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 ++
32 0032 1
33 0033 1 FACILITY: Mount Utility Structure Level 2
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 This module updates the volume set list and home blocks to bind a
38 0038 1 new volume into a volume set.
39 0039 1
40 0040 1 ENVIRONMENT:
41 0041 1
42 0042 1 STARLET operating system, including privileged system services
43 0043 1 and internal exec routines.
44 0044 1
45 0045 1 --
46 0046 1
47 0047 1
48 0048 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 27-Oct-1978 17:54
49 0049 1
50 0050 1 MODIFIED BY:
51 0051 1
52 0052 1 V03-003 HH0041 Hai Huang 24-Jul-1984
53 0053 1 Remove REQUIRE 'SRCDS:[VMSLIB.OBJ]MOUNTMSG.B32'.
54 0054 1
55 0055 1 V03-002 HH0030 Hai Huang 03-Jul-1984
56 0056 1 Remove obsolete LOCK/UNLOCK_INDEXF routines.
57 0057 1
```



```

: 58      0058 1  !      V03-001 STJ0312      Steven T. Jeffreys,      01-Jul-1982
: 59      0059 1  !      Make the code ast reentrant.  This will no longer be
: 60      0060 1  !      necessary once $QIOW is fixed up, but this must be
: 61      0061 1  !      in place for the 3.1 update.
: 62      0062 1  !
: 63      0063 1  !      V02-004 STJ0196      Steven T. Jeffreys,      02-Feb-1982
: 64      0064 1  !      Zero OWN and GLOBAL storage to guaranty restartability.
: 65      0065 1  !      Also make use of a global buffer.
: 66      0066 1  !
: 67      0067 1  !      V02-003 ACG0167      Andrew C. Goldstein,      18-Apr-1980  13:37
: 68      0068 1  !      Previous revision history moved to MOUNT.REV
: 69      0069 1  !**
: 70      0070 1  !
: 71      0071 1  !
: 72      0072 1  !      LIBRARY 'SYSS$LIBRARY:LIB.L32';
: 73      0073 1  !      REQUIRE 'SRCS:MOUDEF.B32';
: 74      0605 1  !
: 75      0606 1  !
: 76      0607 1  !      FORWARD ROUTINE
: 77      0608 1  !      BIND VOLUME      : NOVALUE,      ! main bind routine
: 78      0609 1  !      UPDATE HOME BLK  : NOVALUE,      ! update home blocks on volume
: 79      0610 1  !      OPEN FILE      : NOVALUE,      ! open a file
: 80      0611 1  !      CLOSE FILE     : NOVALUE,      ! close a file
: 81      0612 1  !      READ VIRTUAL   : NOVALUE,      ! read a virtual block
: 82      0613 1  !      WRITE VIRTUAL  : NOVALUE,      ! write a virtual block
: 83      0614 1  !      BIND_HANDLER;   ! condition handler for this module

```



```

: 85      0615 1  !+
: 86      0616 1  !-
: 87      0617 1  ! Global and own storage for this module.
: 88      0618 1  !-
: 89      0619 1  !-
: 90      0620 1  !-
: 91      0621 1  EXTERNAL
: 92      0622 1  CHANNEL,      : BBLOCK,      : channel assigned to device being mounted
: 93      0623 1  BUFFER        : BBLOCK;      : general purpose I/O buffer
: 94      0624 1  HOME_BLOCK    : BBLOCK;      : buffer containing home block of volume
: 95      0625 1  !-
: 96      0626 1  OWN
: 97      0627 1  OWN_START      : VECTOR [0],  : Mark start of OWN storage
: 98      0628 1  CHANNEL2       : WORD,        : channel for index file on RVN 1
: 99      0629 1  CHANNEL3       : WORD,        : channel for volume set list on RVN 1
: 100     0630 1  LOCK_COUNT      :              : saved lock count of volume's index file
: 101     0631 1  LOCK_COUNT1,    :              : as above, for RVN 1
: 102     0632 1  OWN_END        : VECTOR [0];  : Markk end of OWN storage
: 103     0633 1  !-
: 104     0634 1  LITERAL
: 105     0635 1  OWN_LENGTH      = OWN_END - OWN_START;

```

```

107 0636 1 GLOBAL ROUTINE BIND_VOLUME : NOVALUE =
108 0637 1
109 0638 1 !++
110 0639 1
111 0640 1 FUNCTIONAL DESCRIPTION:
112 0641 1
113 0642 1 This routine incorporates a new volume into a volume set. It enters
114 0643 1 it into the volume set list and updates the home blocks on the new
115 0644 1 volume and the root volume.
116 0645 1
117 0646 1
118 0647 1 CALLING SEQUENCE:
119 0648 1 BIND_VOLUME ()
120 0649 1
121 0650 1 INPUT PARAMETERS:
122 0651 1 NONE
123 0652 1
124 0653 1 IMPLICIT INPUTS:
125 0654 1 NONE
126 0655 1
127 0656 1 OUTPUT PARAMETERS:
128 0657 1 NONE
129 0658 1
130 0659 1 IMPLICIT OUTPUTS:
131 0660 1 NONE
132 0661 1
133 0662 1 ROUTINE VALUE:
134 0663 1 NONE
135 0664 1
136 0665 1 SIDE EFFECTS:
137 0666 1 NONE
138 0667 1
139 0668 1 --
140 0669 1
141 0670 2 BEGIN
142 0671 2
143 0672 2 LITERAL
144 0673 2 ENTRY_COUNT = 512 / VSL$C_LENGTH; ! Number of entries per block
145 0674 2 ! in volume set list file
146 0675 2
147 0676 2 LOCAL
148 0677 2 PROCESS PRIV : BBLOCK [8], ! privileges of process
149 0678 2 CLEAR PRIV : BBLOCK [8], ! privilege bits to clear
150 0679 2 NEW SET, ! flag indicating creation of new volume set
151 0680 2 STATUS, ! catch-all status value
152 0681 2 IO_STATUS : VECTOR [4, WORD], ! I/O status block
153 0682 2 EOF, ! last block used in volume set list
154 0683 2 REC_ATTR : BBLOCK [ATTR$S_RECATTR], ! record attributes of vol set list
155 0684 2 P : REF BBLOCK, ! block scan pointer
156 0685 2 FIB : BBLOCK [FIB$C_EXTDATA], ! FIB to extend file
157 0686 2 FIB_DESC : VECTOR [2]; ! descriptor for FIB
158 0687 2
159 0688 2 EXTERNAL
160 0689 2 PHYS_NAME : VECTOR; ! descriptor of physical device name
161 0690 2
162 0691 2
163 0692 2 ENABLE BIND_HANDLER;

```



```

164 0693 2
165 0694 2 ! Zero the OWN and GLOBAL storage so that $MOUNT
166 0695 2 ! my be called repeatedly from a given image.
167 0696 2
168 0697 2
169 0698 2 CH$FILL (0, OWN_LENGTH, OWN_START);
170 0699 2
171 0700 2 ! See if the process has BYPASS and/or SYSPRV privileges. Clear the image
172 0701 2 ! privileges if not, to let file protection work in the /BIND processing.
173 0702 2
174 0703 2
175 0704 2 CLEAR PRIV<0,32> = 0;
176 0705 2 (CLEAR PRIV+4)<0,32> = 0;
177 P 0706 2 $SETPRV (ENBFLG = 0, ! read process privileges
178 P 0707 2 PRMFLG = 1,
179 P 0708 2 PRVADR = CLEAR PRIV,
180 P 0709 2 PRVPRV = PROCESS_PRIV
181 0710 2 );
182 0711 2
183 0712 2 IF NOT .PROCESS PRIV[PRV$V SYSPRV]
184 0713 2 THEN CLEAR PRIV[PRV$V SYSPRV] = 1;
185 0714 2 IF NOT .PROCESS PRIV[PRV$V BYPASS]
186 0715 2 THEN CLEAR PRIV[PRV$V BYPASS] = 1;
187 P 0716 2 $SETPRV (ENBFLG = 0, ! disable image privileges
188 P 0717 2 PRMFLG = 0,
189 P 0718 2 PRVADR = CLEAR_PRIV
190 0719 2 );
191 0720 2
192 0721 2 ! Get a flag indicating whether we are adding a volume to a set or creating
193 0722 2 ! RVN 1 of a new set. This affects various actions along the way.
194 0723 2
195 0724 2
196 0725 2 NEW_SET = .HOME_BLOCK[HM2$W_RVN] EQL 1;
197 0726 2
198 0727 2 ! We already have one channel open on the new volume, which will be used to
199 0728 2 ! access its index file. Open two more channels, for the index file and volume
200 0729 2 ! set list on the root volume. File protection is used to control the user's
201 0730 2 ! privilege to bind a volume; we open all three files concurrently to avoid
202 0731 2 ! error cleanup problems later.
203 0732 2
204 0733 2
205 P 0734 2 STATUS = $ASSIGN (CHAN = CHANNEL2,
206 0735 2 DEVNAM = PHYS_NAME[0]);
207 0736 2 IF NOT .STATUS THEN ERR_EXIT (.STATUS);
208 0737 2
209 P 0738 2 STATUS = $ASSIGN (CHAN = CHANNEL3,
210 0739 2 DEVNAM = PHYS_NAME[0]);
211 0740 2 IF NOT .STATUS THEN ERR_EXIT (.STATUS);
212 0741 2
213 0742 2
214 0743 2 ! Since Version 2, we always mount the volume with the volume "unlocked".
215 0744 2 ! Thus the following call to UNLOCK_INDEX is obsolete. The code is commented
216 0745 2 ! out for historical reasons.
217 0746 2
218 0747 2 ! Patch off the write locks on the index files, so that we can write access
219 0748 2 ! them.
220 0749 2

```

```

221 0750 2 ! STATUS = KERNEL CALL (UNLOCK_INDEXF);
222 0751 2 ! IF NOT .STATUS THEN ERR_EXIT- (.STATUS);
223 0752 2 !
224 0753 2 !
225 0754 2 ! Now open the files.
226 0755 2 !
227 0756 2
228 0757 2 OPEN_FILE (UPLIT WORD (FID$C_INDEXF, FID$C_INDEXF, 0), .CHANNEL, 0);
229 0758 2
230 0759 2 IF NOT .NEW_SET
231 0760 2 THEN OPEN_FILE (UPLIT WORD (FID$C_INDEXF, FID$C_INDEXF, 1), .CHANNEL2, 0);
232 0761 2
233 0762 2 OPEN_FILE (UPLIT WORD (FID$C_VOLSET, FID$C_VOLSET, 1), .CHANNEL3, REC_ATTR);
234 0763 2
235 0764 2 ! We now scan the volume set list file and check for uniqueness of volume
236 0765 2 ! labels.
237 0766 2 !
238 0767 2
239 0768 2 IF CH$EQL (HM2$$_STRUCNAME, HOME_BLOCK[HM2$T_STRUCNAME],
240 0769 2 HM2$$_STRUCNAME, HOME_BLOCK[HM2$T_VOLNAME], ' ')
241 0770 2 THEN ERR_EXIT (MOUNS_DUPVOLNAM);
242 0771 2
243 0772 2 INCR J FROM 1 TO (.HOME_BLOCK[HM2$W_RVN]-1+ENTRY_COUNT-1) / ENTRY_COUNT
244 0773 2 DO
245 0774 2 BEGIN
246 0775 2 READ_VIRTUAL (.CHANNEL3, .J);
247 0776 2
248 0777 2 P = BUFFER;
249 0778 2 INCR K FROM 1 TO ENTRY_COUNT
250 0779 2 DO
251 0780 2 BEGIN
252 0781 2 IF CH$EQL (HM2$$_VOLNAME, HOME_BLOCK[HM2$T_VOLNAME],
253 0782 2 VSL$$_NAME, P[VSL$T_NAME], ' ')
254 0783 2 THEN ERR_EXIT (MOUNS_DUPVOLNAM);
255 0784 2 P = .P + VSL$C_LENGTH;
256 0785 2 END;
257 0786 2
258 0787 2 END;
259 0788 2 ! Enter the new volume in the volume set list and rewrite the block. We
260 0789 2 ! extend the file if necessary.
261 0790 2 !
262 0791 2
263 0792 2 EOF = (.HOME_BLOCK[HM2$W_RVN] + ENTRY_COUNT - 1) / ENTRY_COUNT;
264 0793 2 IF .EOF GEQU .REC_ATTR[FAT$L_EFBLK]
265 0794 2 THEN
266 0795 2 BEGIN
267 0796 2 CH$FILL (0, 512, BUFFER);
268 0797 2 REC_ATTR[FAT$L_EFBLK] = .EOF + 1;
269 0798 2 IF .EOF GTRU .REC_ATTR[FAT$L_HIBLK]
270 0799 2 THEN
271 0800 2 BEGIN
272 0801 2 CH$FILL (0, FIB$C_EXTDATA, FIB);
273 0802 2 FIB[FIB$V_EXTEND] = 1;
274 0803 2 FIB[FIB$V_NOHDREXT] = 1;
275 0804 2 FIB[FIB$L_EXSZ] = 1;
276 0805 2 FIB_DESC[0] = FIB$C_EXTDATA;
277 0806 2 FIB_DESC[1] = FIB;

```



```

278 P 0807 4 STATUS = DO_IO (CHAN = .CHANNEL3,
279 P 0808 4 FUNC = IO$ MODIFY,
280 P 0809 4 IOSB = IO STATUS,
281 P 0810 4 P1 = FIB_DESC
282 0811 4 );
283 0812 4 IF NOT .STATUS THEN ERR EXIT (.STATUS);
284 0813 4 IF NOT .IO STATUS[0] THEN ERR EXIT (.IO STATUS[0]);
285 0814 4 REC_ATTR[FAT$L_HIBLK] = .FIB[FIB$L_EXSZ] + .FIB[FIB$L_EXVBN] - 1;
286 0815 3 END;
287 0816 2 END;
288 0817 2
289 0818 2 P = (.HOME_BLOCK[HM2$W_RVN] MOD ENTRY COUNT) * VSL$C LENGTH + BUFFER;
290 0819 2 CH$MOVE (HM2$S_VOLNAME, HOME_BLOCK[HM2$T_VOLNAME], P[VSL$T_NAME]);
291 0820 2 IF .NEW SET
292 0821 2 THEN CH$MOVE (HM2$S_STRUCNAME, HOME_BLOCK[HM2$T_STRUCNAME], BUFFER[VSL$T_NAME]);
293 0822 2 WRITE_VIRTUAL (.CHANNEL3, .EOF);
294 0823 2
295 0824 2 CLOSE_FILE (.CHANNEL3, REC_ATTR);
296 0825 2
297 0826 2 ! Now update the home blocks on the volumes. On the new volume, we insert the
298 0827 2 ! volume set name and RVN. On RVN 1, we update the count of volumes in the set.
299 0828 2 ! On a new set, both happen on the same volume.
300 0829 2 !
301 0830 2
302 0831 2 IF .NEW_SET
303 0832 2 THEN
304 0833 2 UPDATE_HOME_BLK (.CHANNEL, 3)
305 0834 2 ELSE
306 0835 2 BEGIN
307 0836 2 UPDATE_HOME_BLK (.CHANNEL, 1);
308 0837 2 UPDATE_HOME_BLK (.CHANNEL2, 2);
309 0838 2 END;
310 0839 2
311 0840 2 CLOSE_FILE (.CHANNEL, 0);
312 0841 2 IF NOT .NEW_SET THEN CLOSE_FILE (.CHANNEL2, 0);
313 0842 2
314 0843 2 !
315 0844 2 ! KERNEL_CALL (LOCK_INDEXF);
316 0845 2 !
317 0846 2
318 0847 2 $DASSGN (CHAN = .CHANNEL2);
319 0848 2 $DASSGN (CHAN = .CHANNEL3);
320 0849 2
321 0850 2 ! Re-enable image privileges for the next volume mounted.
322 0851 2 !
323 0852 2
324 P 0853 2 $SETPRV (ENBFLG = 1, ! disable image privileges
325 P 0854 2 PRMFLG = 0,
326 P 0855 2 PRVADR = CLEAR_PRIV
327 0856 2 );
328 0857 2
329 0858 1 END; ! end of routine BIND_VOLUME

```

.TITLE BINDVL
.IDENT \V04-000\

```

.PSECT $SPLITS,NOWRT,NOEXE,2
0000 0001 0001 00000 P.AAA: .WORD 1, 1, 0
0001 0001 0001 00006 P.AAB: .WORD 1, 1, 1
0001 0006 0006 0000C P.AAC: .WORD 6, 6, 1
.PSECT $OWNS,NOEXE,2
00000 OWN_START:
00000 CHANNEL2: .BLKB 0
00002 CHANNEL3: .BLKB 2
00004 LOCK_COUNT: .BLKB 2
00008 LOCK_COUNT1: .BLKB 4
0000C OWN_END: .BLKB 4
0
.EXTRN CHANNEL, BUFFER
.EXTRN HOME_BLOCK, PHYS_NAME
.EXTRN SYSS$SETPRV, SYSS$ASSIGN
.EXTRN COMMON_IO, SYSS$DASSGN
.PSECT $CODE$,NOWRT,2
.ENTRY BIND VOLUME, Save R2,R3,R4,R5,R6,R7,R8,R9,- R10,R11
MOVAB LIB$STOP, R11
MOVAB CHANNEL3, R10
MOVAB -96(SP), SP
MOVAL 19$, (FP)
MOVCS #0, (SP), #0, #12, OWN_START
CLRQ CLEAR_PRIV
PUSHAB PROCESS_PRIV
PUSHL #1
PUSHAB CLEAR_PRIV
CLRL -(SP)
CALLS #4, SYSS$SETPRV
BBS #4, PROCESS_PRIV+3, 1$
BISB2 #16, CLEAR_PRIV+3
BBS #5, PROCESS_PRIV+3, 2$
BISB2 #32, CLEAR_PRIV+3
CLRQ -(SP)
PUSHAB CLEAR_PRIV
CLRL -(SP)
CALLS #4, SYSS$SETPRV
CLRL R0
CMPW HOME_BLOCK+38, #1
BNEQ 3$
INCL R0
MOVL R0, NEW_SET
CLRQ -(SP)
PUSHAB CHANNEL2
PUSHAB PHYS_NAME

```

OC

00

04

04

```

OFFC 00000
5B 00000000G 00 9E 00002
5A 0000' CF 9E 00009
5E A0 AE 9E 0000E
6D 021A CF DE 00012
6E 00 2C 00017
FE AA 0001C
50 AE 7C 0001E
58 AE 9F 00021
01 DD 00024
58 AE 9F 00026
7E D4 00029
04 FB 0002B
04 E0 00032
10 88 00037
05 E0 0003B 1$:
20 88 00040
7E 7C 00044 2$:
58 AE 9F 00046
7E D4 00049
04 FB 0004B
50 D4 00052
01 0000G CF B1 00054
02 12 00059
50 D6 0005B
50 D0 0005D 3$:
59 7E 7C 00060
FE AA 9F 00062
0000G CF 9F 00065

```

:

0636

0670

0698

0704

0710

0712

0713

0714

0715

0719

0725

0735

00000000G	00	04	FB	00069	CALLS	#4, SYSS\$ASSIGN	:	
	57	50	DO	00070	MOVL	R0, STATUS	:	
	05	57	E8	00073	BLBS	STATUS, 4\$:	0736
	6B	57	DD	00076	PUSHL	STATUS	:	
		01	FB	00078	CALLS	#1, LIB\$STOP	:	
		7E	7C	0007B	4\$: CLRQ	-(SP)	:	0739
		5A	DD	0007D	PUSHL	R10	:	
		CF	9F	0007F	PUSHAB	PHYS NAME	:	
00000000G	00	04	FB	00083	CALLS	#4, SYSS\$ASSIGN	:	
	57	50	DO	0008A	MOVL	R0, STATUS	:	
	05	57	E8	0008D	BLBS	STATUS, 5\$:	0740
	6B	57	DD	00090	PUSHL	STATUS	:	
		01	FB	00092	CALLS	#1, LIB\$STOP	:	
		7E	D4	00095	5\$: CLRL	-(SP)	:	0757
		CF	DD	00097	PUSHL	CHANNEL	:	
		CF	9F	0009B	PUSHAB	P.AAA	:	
0000V	CF	03	FB	0009F	CALLS	#3, OPEN_FILE	:	
	0F	59	E8	000A4	BLBS	NEW SET, 6\$:	0759
		7E	D4	000A7	CLRL	-(SP)	:	0760
	7E	FE	AA	000A9	MOVZWL	CHANNEL2, -(SP)	:	
		CF	9F	000AD	PUSHAB	P.AAB	:	
0000V	CF	03	FB	000B1	CALLS	#3, OPEN_FILE	:	
		AE	9F	000B6	6\$: PUSHAB	REC ATTR	:	0762
	7E	6A	3C	000B9	MOVZWL	CHANNEL3, -(SP)	:	
		CF	9F	000BC	PUSHAB	P.AAC	:	
0000G	CF	03	FB	000C0	CALLS	#3, OPEN_FILE	:	
	0000G	0C	29	000C5	CMPC3	#12, HOME_BLOCK+460, HOME_BLOCK+472	:	0769
		09	12	000CD	BNEQ	7\$:	
	6B	007281AC	8F	DD	000CF	PUSHL	#7504300	0770
	56	0000G	01	FB	000D5	CALLS	#1, LIB\$STOP	
	56		CF	3C	000D8	7\$: MOVZWL	HOME_BLOCK+38, R6	0772
	56		06	C0	000DD	ADDL2	#6, R6	
			08	C6	000E0	DIVL2	#8, R6	
			54	D4	000E3	CLRL	J	
			2B	11	000E5	BRB	11\$	
			54	DD	000E7	8\$: PUSHL	J	0775
	7E		6A	3C	000E9	MOVZWL	CHANNEL3, -(SP)	
0000V	CF	02	FB	000EC	CALLS	#2, READ_VIRTUAL	:	
	58	0000G	CF	9E	000F1	MOVAB	BUFFER, P	0777
	55		01	DO	000F6	MOVL	#1, K	0778
68	0000G	CF	0C	29	000F9	9\$: CMPC3	#12, HOME_BLOCK+472, (P)	0782
			09	12	000FF	BNEQ	10\$	
		007281AC	8F	DD	00101	PUSHL	#7504300	0783
	6B		01	FB	00107	CALLS	#1, LIB\$STOP	
	58	40	A8	9E	0010A	10\$: MOVAB	64(R8), P	0784
E7			08	F3	0010E	AOBLEQ	#8, K, 9\$	0778
D1			56	F3	00112	11\$: AOBLEQ	R6, J, 8\$	0772
		0000G	CF	3C	00116	MOVZWL	HOME_BLOCK+38, R0	0792
			07	C0	0011B	ADDL2	#7, R0	
56			08	C7	0011E	DIVL3	#8, R0, EOF	
	30		56	D1	00122	CMPL	EOF, REC_ATTR+8	0793
			6B	1F	00126	BLSSU	14\$	
0200	8F	00	00	2C	00128	MOVCS	#0, (SP), #0, #512, BUFFER	0796
			CF		0012F			
	30		A6	9E	00132	MOVAB	1(R6), REC_ATTR+8	0797
	2C		56	D1	00137	CMPL	EOF, REC_ATTR+4	0798
			56	1B	0013B	BLEQU	14\$	

20	00	6E	00	2C	0013D	MOVC5	#0, (SP), #0, #32, FIB	: 0801
		08	AE		00142			
	1E	AE	0280	8F	A8 00144	BISW2	#640, FIB+23	: 0803
	20	AE		01	D0 0014A	MOVL	#1, FIB+24	: 0804
		6E		20	D0 0014E	MOVL	#32, FIB_DESC	: 0805
	04	AE	08	AE	9E 00151	MOVAB	FIB, FIB_DESC+4	: 0806
				7E	7C 00156	CLRQ	-(SP)	: 0811
				7E	7C 00158	CLRQ	-(SP)	
				7E	D4 0015A	CLRL	-(SP)	
			14	AE	9F 0015C	PUSHAB	FIB_DESC	
				7E	7C 0015F	CLRQ	-(SP)	
			68	AE	9F 00161	PUSHAB	IO_STATUS	
		7E		36	DD 00164	PUSHL	#54	
				6A	3C 00166	MOVZWL	CHANNEL3, -(SP)	
	00000000G	00		1A	DD 00169	PUSHL	#26	
		57		0C	FB 0016B	CALLS	#12, COMMON_IO	
		05		50	D0 00172	MOVL	R0, STATUS	
				57	E8 00175	BLBS	STATUS, 12\$: 0812
				57	DD 00178	PUSHL	STATUS	
		6B		01	FB 0017A	CALLS	#1, LIB\$STOP	
		07	48	AE	E8 0017D	BLBS	IO_STATUS, 13\$: 0813
		7E	48	AE	3C 00181	MOVZWL	IO_STATUS, -(SP)	
		6B		01	FB 00185	CALLS	#1, LIB\$STOP	
	50	AE	24	AE	C1 00188	ADDL3	FIB+28, FIB+24, R0	: 0814
		2C	FF	A0	9E 0018E	MOVAB	-1(R0), REC_ATTR+4	
		50	0000G	CF	3C 00193	MOVZWL	HOME_BLOCK+38, R0	: 0818
		50		01	7A 00198	EMUL	#1, R0, #0, -(SP)	
7E	00	8E		08	7B 0019D	EDIV	#8, (SP)+, R0, R0	
50	50	50		06	78 001A2	ASHL	#6, R0, R0	
		58	0000GCF	40	9E 001A6	MOVAB	BUFFER[R0], P	
	68	0000G		0C	28 001AC	MOVC3	#12, HOME_BLOCK+472, (P)	: 0819
		08		59	E9 001B2	BLBC	NEW_SET, T5\$: 0820
0000G	CF	0000G	CF	0C	28 001B5	MOVC3	#12, HOME_BLOCK+460, BUFFER	: 0821
				56	DD 001BD	PUSHL	EOF	: 0822
		7E		6A	3C 001BF	MOVZWL	CHANNEL3, -(SP)	
	0000V	CF		02	FB 001C2	CALLS	#2, WRITE_VIRTUAL	
			28	AE	9F 001C7	PUSHAB	REC_ATTR	: 0824
		7E		6A	3C 001CA	MOVZWL	CHANNEL3, -(SP)	
	0000V	CF		02	FB 001CD	CALLS	#2, CLOSE_FILE	
		08		59	E9 001D2	BLBC	NEW_SET, T6\$: 0833
			0000G	03	DD 001D5	PUSHL	#3	
				CF	DD 001D7	PUSHL	CHANNEL	
				11	11 001DB	BRB	17\$	
				01	DD 001DD	PUSHL	#1	: 0836
			0000G	CF	DD 001DF	PUSHL	CHANNEL	
	0000V	CF		02	FB 001E3	CALLS	#2, UPDATE_HOME_BLK	
				02	DD 001E8	PUSHL	#2	: 0837
		7E	FE	AA	3C 001EA	MOVZWL	CHANNEL2, -(SP)	
	0000V	CF		02	FB 001EE	CALLS	#2, UPDATE_HOME_BLK	
				7E	D4 001F3	CLRL	-(SP)	: 0840
			0000G	CF	DD 001F5	PUSHL	CHANNEL	
	0000V	CF		02	FB 001F9	CALLS	#2, CLOSE_FILE	
		0B		59	E8 001FE	BLBS	NEW_SET, T8\$: 0841
		7E		7E	D4 00201	CLRL	-(SP)	
			FE	AA	3C 00203	MOVZWL	CHANNEL2, -(SP)	
	0000V	CF		02	FB 00207	CALLS	#2, CLOSE_FILE	
		7E	FE	AA	3C 0020C	MOVZWL	CHANNEL2, -(SP)	: 0847

BINDVL
V04-000

M 16
16-Sep-1984 01:10:14
14-Sep-1984 12:45:16

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[MOUNT.SRC]BINDVL.B32;1

Page 11
(3)

00000000G	00		01	FB	00210	CALLS	#1, SYSSDASSGN	:	
	7E		6A	3C	00217	MOVZWL	CHANNEL3, -(SP)	:	0848
00000000G	00		01	FB	0021A	CALLS	#1, SYSSDASSGN	:	
			7E	7C	00221	CLRQ	-(SP)	:	0856
		58	AE	9F	00223	PUSHAB	CLEAR_PRIV	:	
			01	DD	00226	PUSHL	#1	:	
00000000G	00		04	FB	00228	CALLS	#4, SYSSSETPRV	:	
				04	0022F	RET		:	0858
				0000	00230	19\$:	.WORD	:	0670
			7E	D4	00232	CLRL	-(SP)	:	
			5E	DD	00234	PUSHL	SP	:	
	7E		AC	7D	00236	MOVQ	4(AP), -(SP)	:	
0000V	CF		03	FB	0023A	CALLS	#3, BIND_HANDLER	:	
			04	0023F	RET			:	

; Routine Size: 576 bytes, Routine Base: \$CODE\$ + 0000


```

: 331 0859 1 ROUTINE UPDATE_HOME_BLK (CHANNEL, MODE) : NOVALUE =
: 332 0860 1
: 333 0861 1 !++
: 334 0862 1
: 335 0863 1 FUNCTIONAL DESCRIPTION:
: 336 0864 1
: 337 0865 1 This routine updates the home blocks in the index file open on the
: 338 0866 1 indicated channel. It enters volume set name, RVN, and number of
: 339 0867 1 volumes as directed.
: 340 0868 1
: 341 0869 1
: 342 0870 1 CALLING SEQUENCE:
: 343 0871 1 UPDATE_HOME_BLK (ARG1, ARG2)
: 344 0872 1
: 345 0873 1 INPUT PARAMETERS:
: 346 0874 1 ARG1: channel on which index file is open
: 347 0875 1 ARG2: mode flag:
: 348 0876 1 bit 0 set to update RVN and structure name
: 349 0877 1 bit 1 set to update count of volumes in set
: 350 0878 1
: 351 0879 1 IMPLICIT INPUTS:
: 352 0880 1 HOME_BLOCK: home block of volume being mounted, containing needed data
: 353 0881 1
: 354 0882 1 OUTPUT PARAMETERS:
: 355 0883 1 NONE
: 356 0884 1
: 357 0885 1 IMPLICIT OUTPUTS:
: 358 0886 1 NONE
: 359 0887 1
: 360 0888 1 ROUTINE VALUE:
: 361 0889 1 NONE
: 362 0890 1
: 363 0891 1 SIDE EFFECTS:
: 364 0892 1 index file of specified disk written
: 365 0893 1
: 366 0894 1 --
: 367 0895 1
: 368 0896 2 BEGIN
: 369 0897 2
: 370 0898 2 MAP
: 371 0899 2 MODE : BITVECTOR; ! mode flags arg
: 372 0900 2
: 373 0901 2 LOCAL
: 374 0902 2 ERR_COUNT, ! inverse count of errors encountered
: 375 0903 2 COUNT, ! highest VBN to process
: 376 0904 2 VBN, ! current VBN in index file
: 377 0905 2 STATUS, ! catch-all status value
: 378 0906 2 STATUS2, ! 2nd status value
: 379 0907 2 IO_STATUS : VECTOR [4, WORD]; ! I/O status block
: 380 0908 2
: 381 0909 2 EXTERNAL ROUTINE
: 382 0910 2 CHECK_HOME_BLK2, ! verify level 2 home block
: 383 0911 2 CHECKSUM2; ! compute home block checksum
: 384 0912 2
: 385 0913 2
: 386 0914 2 ! We read and update all of the home blocks of the volume. Each home block, as
: 387 0915 2 ! it is read, is checked for validity. If there is an error, we write back that

```



```

388 0916 2 ! home block with a bad checksum to prevent misinterpretation of bad data.
389 0917 2 ! On a second such error, we give up to avoid risking leaving a volume with
390 0918 2 ! no good home blocks.
391 0919 2
392 0920 2
393 0921 2 ERR COUNT = 2; ! we quit after the 2nd error
394 0922 2 COUNT = -1;
395 0923 2 VBN = 2;
396 0924 2 DO
397 0925 2 BEGIN
398 0926 2 STATUS = DO_IO (CHAN = .CHANNEL,
399 0927 2 FUNC = IOS_READVBLK,
400 0928 2 IOSB = IO_STATUS,
401 0929 2 P1 = BUFFER,
402 0930 2 P2 = 512,
403 0931 2 P3 = .VBN
404 0932 2 );
405 0933 2 IF .STATUS THEN STATUS = .IO_STATUS[0];
406 0934 2 IF .STATUS
407 0935 2 THEN IF NOT CHECK_HOME_BLK2 (BUFFER, .BUFFER[HM2$L_HOME_LBN],
408 0936 2 UPLIT (HM2$S_VOLNAME, BUFFER[HM2$T_VOLNAME]))
409 0937 2 THEN STATUS = MOUN$_HOMBLKCHK;
410 0938 2
411 0939 2 IF NOT .STATUS
412 0940 2 THEN
413 0941 2 BEGIN
414 0942 2 ERR_COUNT = .ERR_COUNT - 1;
415 0943 2 IF .ERR_COUNT LEQ 0
416 0944 2 THEN ERR_EXIT (MOUN$_BADHOMBLK, 0, .STATUS);
417 0945 2
418 0946 2 ERR_MESSAGE (MOUN$_BADHOMBLK, 0, .STATUS);
419 0947 2 END
420 0948 2
421 0949 2 ELSE ! get loop count from 1st good home block
422 0950 2 IF .COUNT EQL -1 THEN COUNT = .BUFFER[HM2$W_CLUSTER] * 3;
423 0951 2
424 0952 2 ! Update the home block read with structure name, RVN, and/or volume set size
425 0953 2 ! as requested. Recompute the checksums. If the block was bad, bash a checksum.
426 0954 2 ! Finally rewrite it.
427 0955 2
428 0956 2
429 0957 2 IF .MODE[0]
430 0958 2 THEN
431 0959 2 BEGIN
432 0960 2 CH$MOVE (HM2$S_STRUCNAME, HOME_BLOCK[HM2$T_STRUCNAME], BUFFER[HM2$T_STRUCNAME]);
433 0961 2 BUFFER[HM2$W_RVN] = .HOME_BLOCK[HM2$W_RVN];
434 0962 2 END;
435 0963 2
436 0964 2 IF .MODE[1]
437 0965 2 THEN
438 0966 2 BUFFER[HM2$W_SETCOUNT] = .HOME_BLOCK[HM2$W_RVN];
439 0967 2
440 0968 2 CHECKSUM2 (BUFFER, $BYTEOFFSET (HM2$W_CHECKSUM1));
441 0969 2 CHECKSUM2 (BUFFER, $BYTEOFFSET (HM2$W_CHECKSUM2));
442 0970 2 IF NOT .STATUS THEN BUFFER[HM2$W_CHECKSUM2] = NOT .BUFFER[HM2$W_CHECKSUM2];
443 0971 2
444 0972 2 STATUS2 = DO_IO (CHAN = .CHANNEL,

```



```

: 445 P 0973 3 FUNC = IO$ WRITEVBLK,
: 446 P 0974 3 IOSB = IO STATUS,
: 447 P 0975 3 P1 = BUFFER,
: 448 P 0976 3 P2 = 512,
: 449 P 0977 3 P3 = .VBN
: 450 0978 3 );
: 451 0979 3 IF .STATUS2 THEN STATUS2 = .IO_STATUS[0];
: 452 0980 3
: 453 0981 3 IF .STATUS ! ignore write error if it was bad
: 454 0982 3 THEN
: 455 0983 3 IF NOT .STATUS2
: 456 0984 3 THEN
: 457 0985 4 BEGIN
: 458 0986 4 ERR_MESSAGE (MOUN$ WRTHOMBLK, 0, .STATUS2);
: 459 0987 4 ERR_COUNT = .ERR_COUNT - 1;
: 460 0988 4 END;
: 461 0989 3
: 462 0990 3 VBN = .VBN + 1;
: 463 0991 3 END
: 464 0992 2 UNTIL .VBN GTRU .COUNT; ! loop for all home blocks
: 465 0993 2
: 466 0994 1 END; ! end of routine UPDATE_HOME_BLK

```

```

.PSECT $PLITS$,NOWRT,NOEXE,2
0000000C 00012 P.AAD: .BLKB 2
00000000G 00014 .LONG 12
00000000G 00018 .ADDRESS BUFFER+472
.EXTRN CHECK_HOME_BLK2, CHECKSUM2
.PSECT $CODE$,NOWRT,2

```

```

OFFC 00000 UPDATE_HOME_BLK:
5B 0000G CF 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 : 0859
5E 08 C2 00007 MOVAB BUFFER, R11
5A 02 D0 0000A SUBL2 #8, SP
59 01 CE 0000D MOVL #2, ERR_COUNT : 0921
57 02 D0 00010 MNEGL #1, COUNT : 0922
7E 7E 7C 00013 1$: MOVL #2, VBN : 0923
7E D4 00015 CLRQ -(SP) : 0932
57 DD 00017 CLRL -(SP)
7E 0200 8F 3C 00019 PUSHL VBN
5B DD 0001E MOVZWL #512, -(SP)
7E 7C 00020 PUSHL R11
20 AE 9F 00022 CLRL -(SP)
31 DD 00025 PUSHAB IO STATUS
04 AC DD 00027 PUSHL #49
1A DD 0002A PUSHL CHANNEL
00000000G 00 0C FB C002C PUSHL #26
56 50 D0 00033 CALLS #12, COMMON_IO
20 56 E9 00036 MOVL R0, STATUS
56 6E 3C 00039 BLBC STATUS, 3$ : 0933
1A 56 E9 0003C MOVZWL IO STATUS, STATUS
BLBC STATUS, 3$ : 0934

```


			0000'	CF	9F	0003F	PUSHAB	P.AAD		0936
				6B	DD	00043	PUSHL	BUFFER		0935
				5B	DD	00045	PUSHL	R11		
	0000G	CF		03	FB	00047	CALLS	#3, CHECK_HOMEBLK2		
		07		50	E8	0004C	BLBS	R0, 2\$		
		56	007281B4	8F	D0	0004F	MOVL	#7504308, STATUS		0937
		27		56	E8	00056	BLBS	STATUS, 5\$		0939
		11		5A	F5	00059	SOBGTR	ERR_COUNT, 4\$		0942
				56	DD	0005C	PUSHL	STATUS		0944
				7E	D4	0005E	CLRL	-(SP)		
			00729028	8F	DD	00060	PUSHL	#7508008		
	00000000G	00		03	FB	00066	CALLS	#3, LIB\$STOP		
				56	DD	0006D	PUSHL	STATUS		0946
				7E	D4	0006F	CLRL	-(SP)		
			00729028	8F	DD	00071	PUSHL	#7508008		
	00000000G	00		03	FB	00077	CALLS	#3, LIB\$SIGNAL		
				10	11	0007E	BRB	6\$		0939
	FFFFFFFF	8F		59	D1	00080	CMPL	COUNT, #-1		0950
				07	12	00087	BNEQ	6\$		
		59	0E	AB	3C	00089	MOVZWL	BUFFER+14, COUNT		
		59		03	C4	0008D	MULL2	#3, COUNT		
		0E	08	AC	E9	00090	BLBC	MODE, 7\$		0957
01CC	CB	0000G		0C	28	00094	MOV3	#12, HOME_BLOCK+460, BUFFER+460		0960
		26	AB	0000G	CF	B0	MOVW	HOME_BLOCK+38, BUFFER+38		0961
	06	08	AC		01	E1	BBC	#1, MODE, 8\$		0964
		28	AB	0000G	CF	B0	MOVW	HOME_BLOCK+38, BUFFER+40		0966
				3A	DD	000AD	PUSHL	#58		0968
				5B	DD	000AF	PUSHL	R11		
	0000G	CF		02	FB	000B1	CALLS	#2, CHECKSUM2		
		7E	01FE	8F	3C	000B6	MOVZWL	#510, -(SP)		0969
				5B	DD	000BB	PUSHL	R11		
	0000G	CF		02	FB	000BD	CALLS	#2, CHECKSUM2		
		07		56	E8	000C2	BLBS	STATUS, 9\$		0970
	01FE	CB	01FE	CB	B2	000C5	MCOMW	BUFFER+510, BUFFER+510		
				7E	7C	000CC	CLRQ	-(SP)		0978
				7E	D4	000CE	CLRL	-(SP)		
				57	DD	000D0	PUSHL	VCN		
		7E	0200	8F	3C	000D2	MOVZWL	#512, -(SP)		
				5B	DD	000D7	PUSHL	R11		
				7E	7C	000D9	CLRQ	-(SP)		
			20	AE	9F	000DB	PUSHAB	IO STATUS		
				30	DD	000DE	PUSHL	#48		
			04	AC	DD	000E0	PUSHL	CHANNEL		
				1A	DD	000E3	PUSHL	#26		
	00000000G	00		0C	FB	000E5	CALLS	#12, COMMON_IO		
		58		50	D0	000EC	MOVL	R0, STATUS2		
		03		58	E9	000EF	BLBC	STATUS2, 10\$		0979
		58		6E	3C	000F2	MOVZWL	IO STATUS, STATUS2		
		16		56	E9	000F5	BLBC	STATUS, 11\$		0981
		13		58	E8	000F8	BLBS	STATUS2, 11\$		0983
				58	DD	000FB	PUSHL	STATUS2		0986
				7E	D4	000FD	CLRL	-(SP)		
			00729030	8F	DD	000FF	PUSHL	#7508016		
	00000000G	00		03	FB	00105	CALLS	#3, LIB\$SIGNAL		
				5A	D7	0010C	DECL	ERR_COUNT		0987
				57	D6	0010E	INCL	VCN		0990
		59		57	D1	00110	CMPL	VCN, COUNT		0992

BINDVL
V04-000

F 1
16-Sep-1984 01:10:14
14-Sep-1984 12:45:16

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[MOUNT.SRC]BINDVL.B32;1 Page 16
(4)

03	1A	00113	BGTRU	12\$
FEFB	31	00115	BRW	1\$
	04	00118	RET	

:
:
: 0994

; Routine Size: 281 bytes, Routine Base: \$CODE\$ + 0240


```

468 0995 1 ROUTINE OPEN_FILE (FID, CHANNEL, ATTRIB) : NOVALUE =
469 0996 1
470 0997 1 ++
471 0998 1
472 0999 1 FUNCTIONAL DESCRIPTION:
473 1000 1
474 1001 1 This routine opens the file given by the file ID on the indicated
475 1002 1 channel.
476 1003 1
477 1004 1
478 1005 1 CALLING SEQUENCE:
479 1006 1 OPEN_FILE (ARG1, ARG2, ARG3)
480 1007 1
481 1008 1 INPUT PARAMETERS:
482 1009 1 ARG1: address of file ID
483 1010 1 ARG2: channel number to use
484 1011 1
485 1012 1 IMPLICIT INPUTS:
486 1013 1 NONE
487 1014 1
488 1015 1 OUTPUT PARAMETERS:
489 1016 1 ARG3: address of buffer to receive record attributes, if not 0
490 1017 1
491 1018 1 IMPLICIT OUTPUTS:
492 1019 1 NONE
493 1020 1
494 1021 1 ROUTINE VALUE:
495 1022 1 NONE
496 1023 1
497 1024 1 SIDE EFFECTS:
498 1025 1 file opened
499 1026 1
500 1027 1 --
501 1028 1
502 1029 2 BEGIN
503 1030 2
504 1031 2 MAP
505 1032 2
506 1033 2 FID : REF BBLOCK, ! file ID arg
507 1034 2 ATTRIB : REF BBLOCK; ! attribute buffer arg
508 1035 2
509 1036 2 BUILTIN
510 1037 2 ROT;
511 1038 2
512 1039 2 LOCAL
513 1040 2 STATUS, ! general status return
514 1041 2 IO_STATUS : VECTOR [4, WORD], ! I/O status block
515 1042 2 ATTR_CTL : BBLOCKVECTOR [2, 8], ! attribute control list
516 1043 2 FIB : BBLOCK [FIB$C_ACCDATA], ! FIB
517 1044 2 FIB_DESC : VECTOR [2]; ! FIB descriptor
518 1045 2
519 1046 2
520 1047 2 ! Fill in the control blocks and open the file.
521 1048 2 !
522 1049 2
523 1050 2 CH$MOVE (FIB$C_LENGTH, .FID, FIB[FIB$W_FID]);
524 1051 2 FIB[FIB$L_ACCT] = FIB$M_WRITE OR FIB$M_NOWRITE;

```

```

525 1052 2
526 1053 2 ATTR_CTL[0, ATR$W_SIZE] = 0;
527 1054 2 ATTR_CTL[0, ATR$W_TYPE] = 0;
528 1055 2 IF .ATTRIB NEQ 0
529 1056 2 THEN
530 1057 2 BEGIN
531 1058 2 ATTR_CTL[0, ATR$W_SIZE] = ATR$S_RECATTR;
532 1059 2 ATTR_CTL[0, ATR$W_TYPE] = ATR$C_RECATTR;
533 1060 2 ATTR_CTL[0, ATR$W_ADDR] = .ATTRIB;
534 1061 2 ATTR_CTL[1, ATR$W_SIZE] = 0;
535 1062 2 ATTR_CTL[1, ATR$W_TYPE] = 0;
536 1063 2 END;
537 1064 2
538 1065 2 FIB_DESC[0] = FIB$C_ACCDATA;
539 1066 2 FIB_DESC[1] = FIB;
540 1067 2
541 P 1068 2 STATUS = DO_IO (CHAN = .CHANNEL,
542 P 1069 2 FUNC = IO$ACCESS OR IO$M_ACCESS,
543 P 1070 2 IOSB = IO STATUS,
544 P 1071 2 P1 = FIB_DESC,
545 P 1072 2 P5 = ATTR_CTL
546 1073 2 );
547 1074 2 IF .STATUS THEN STATUS = .IO_STATUS[0];
548 1075 2 IF NOT .STATUS THEN ERR_EXIT(.STATUS);
549 1076 2
550 1077 2 ! If attributes were requested, invert the numbers into the standard longword
551 1078 2 ! format and normalize the end of file mark.
552 1079 2 !
553 1080 2
554 1081 2 IF .ATTRIB NEQ 0
555 1082 2 THEN
556 1083 2 BEGIN
557 1084 2 ATTRIB[FAT$W_HIBLK] = ROT (.ATTRIB[FAT$W_HIBLK], 16);
558 1085 2 ATTRIB[FAT$W_EFBLK] = ROT (.ATTRIB[FAT$W_EFBLK], 16);
559 1086 2 IF .ATTRIB[FAT$W_FFBYTE] NEQ 0
560 1087 2 THEN
561 1088 2 BEGIN
562 1089 2 ATTRIB[FAT$W_FFBYTE] = 0;
563 1090 2 ATTRIB[FAT$W_EFBLK] = .ATTRIB[FAT$W_EFBLK] + 1;
564 1091 2 END;
565 1092 2 END;
566 1093 2
567 1094 1 END;

```

! end of routine OPEN_FILE

				003C 00000 OPEN_FILE:					
			5E	2C	C2	00002	.WORD	Save R2,R3,R4,R5	0995
OC	AE	04	BC	06	28	00005	SUBL2	#44, SP	1050
		08	AE	8F	3C	0000B	MOVCL	#6, @FID, FIB+4	1051
				AE	D4	00011	MOVZWL	#257, FIB	1053
			52	AC	D0	00014	CLRL	ATTR_CTL	1055
				53	D4	00018	MOVL	ATTRIB, R2	
				52	D5	0001A	CLRL	R3	
							TSTL	R2	

			11	13	0001C	BEQL	1\$:	
			53	D6	0001E	INCL	R3	:	
14	AE	00040020	8F	D0	00020	MOVL	#262176, ATTR_CTL	:	1058
18	AE		52	D0	00028	MOVL	R2, ATTR_CTL+4	:	1060
		1C	AE	D4	0002C	CLRL	ATTR_CTL+8	:	1061
	6E		0A	D0	0002F	MOVL	#10, FIB_DESC	:	1065
04	AE	08	AE	9E	00032	MOVAB	FIB, FIB_DESC+4	:	1066
			7E	D4	00037	CLRL	-(SP)	:	1073
		18	AE	9F	00039	PUSHAB	ATTR_CTL	:	
			7E	7C	0003C	CLRQ	-(SP)	:	
			7E	D4	0003E	CLRL	-(SP)	:	
		14	AE	9F	00040	PUSHAB	FIB_DESC	:	
			7E	7C	00043	CLRQ	-(SP)	:	
		44	AE	9F	00045	PUSHAB	IO STATUS	:	
	7E		72	8F	9A 00048	MOVZBL	#1T4, -(SP)	:	
		08	AC	DD	0004C	PUSHL	CHANNEL	:	
			1A	DD	0004F	PUSHL	#26	:	
00000000G	00		0C	FB	00051	CALLS	#12, COMMON_IO	:	
	07		50	E9	00058	BLBC	STATUS, 2\$:	1074
	50	24	AE	3C	0005B	MOVZWL	IO STATUS, STATUS	:	
	09		50	E8	0005F	BLBS	STATUS, 3\$:	1075
00000000G	00		50	DD	00062	PUSHL	STATUS	:	
	17		01	FB	00064	CALLS	#1, LIB\$STOP	:	
04	A2	04	53	E9	0006B	BLBC	R3, 4\$:	1081
08	A2	08	10	9C	0006E	ROTL	#16, 4(R2), 4(R2)	:	1084
			10	9C	00074	ROTL	#16, 8(R2), 8(R2)	:	1085
		0C	A2	B5	0007A	TSTW	12(R2)	:	1086
			06	13	0007D	BEQL	4\$:	
		0C	A2	B4	0007F	CLRW	12(R2)	:	1089
		08	A2	D6	00082	INCL	8(R2)	:	1090
			04	00085	4\$:	RET		:	1094

; Routine Size: 134 bytes, Routine Base: \$CODE\$ + 0359

```

: 569 1095 1 ROUTINE CLOSE_FILE (CHANNEL, ATTRIB) : NOVALUE =
: 570 1096 1
: 571 1097 1 !++
: 572 1098 1
: 573 1099 1 FUNCTIONAL DESCRIPTION:
: 574 1100 1
: 575 1101 1 This routine closes the file open on the given channel.
: 576 1102 1
: 577 1103 1
: 578 1104 1 CALLING SEQUENCE:
: 579 1105 1 CLOSE_FILE (ARG1, ARG2)
: 580 1106 1
: 581 1107 1 INPUT PARAMETERS:
: 582 1108 1 ARG1: channel number to use
: 583 1109 1 ARG2: address of buffer to receive record attributes, if not 0
: 584 1110 1
: 585 1111 1 IMPLICIT INPUTS:
: 586 1112 1 NONE
: 587 1113 1
: 588 1114 1 OUTPUT PARAMETERS:
: 589 1115 1 NONE
: 590 1116 1
: 591 1117 1 IMPLICIT OUTPUTS:
: 592 1118 1 NONE
: 593 1119 1
: 594 1120 1 ROUTINE VALUE:
: 595 1121 1 NONE
: 596 1122 1
: 597 1123 1 SIDE EFFECTS:
: 598 1124 1 file closed
: 599 1125 1
: 600 1126 1 !--
: 601 1127 1
: 602 1128 2 BEGIN
: 603 1129 2
: 604 1130 2 MAP
: 605 1131 2 ATTRIB : REF BBLOCK; ! attribute buffer arg
: 606 1132 2
: 607 1133 2 BUILTIN
: 608 1134 2 ROT;
: 609 1135 2
: 610 1136 2 LOCAL
: 611 1137 2 STATUS, ! general status return
: 612 1138 2 IO STATUS : VECTOR [4, WORD], ! I/O status block
: 613 1139 2 ATTR_CTL : BBLOCKVECTOR [2, 8]; ! attribute control list
: 614 1140 2
: 615 1141 2
: 616 1142 2 ! Fill in the control blocks and close the file.
: 617 1143 2 !
: 618 1144 2
: 619 1145 2 ATTR_CTL[0, ATR$W_SIZE] = 0;
: 620 1146 2 ATTR_CTL[0, ATR$W_TYPE] = 0;
: 621 1147 2 IF .ATTRIB NEQ 0
: 622 1148 2 THEN
: 623 1149 3 BEGIN
: 624 1150 3 ATTR_CTL[0, ATR$W_SIZE] = ATR$S_RECATTR;
: 625 1151 3 ATTR_CTL[0, ATR$W_TYPE] = ATR$C_RECATTR;

```



```

: 626      1152      3      ATTR_CTL[0, ATR$SL_ADDR] = .ATTRIB;
: 627      1153      3      ATTR_CTL[1, ATR$W_SIZE] = 0;
: 628      1154      3      ATTR_CTL[1, ATR$W_TYPE] = 0;
: 629      1155      3
: 630      1156      3      ATTRIB[FAT$SL_HIBLK] = ROT (.ATTRIB[FAT$SL_HIBLK], 16);
: 631      1157      3      ATTRIB[FAT$SL_EFBLK] = ROT (.ATTRIB[FAT$SL_EFBLK], 16);
: 632      1158      3      END;
: 633      1159      3
: 634      1160      2      STATUS = DO_IO (CHAN = .CHANNEL,
: 635      1161      2      FUNC = IO$ DEACCESS,
: 636      1162      2      IOSB = IO STATUS,
: 637      1163      2      P5 = ATTR_CTL
: 638      1164      2      );
: 639      1165      2      IF .STATUS THEN STATUS = .IO_STATUS[0];
: 640      1166      2      IF NOT .STATUS THEN ERR_EXIT(.STATUS);
: 641      1167      2
: 642      1168      1      END;

```

! end of routine CLOSE_FILE

				0000 00000 CLOSE_FILE:				
		5E		14	C2 00002	.WORD	Save nothing	: 1095
				7E	D4 00005	SUBL2	#20, SP	
		50	08	AC	D0 00007	CLRL	ATTR_CTL	: 1145
				1A	13 0000B	MOVL	ATTRIB, R0	: 1147
		6E	00040020	8F	D0 0000D	BEQL	1\$	
		04	AE	50	D0 00014	MOVL	#262176, ATTR_CTL	: 1150
				AE	D4 00018	MOVL	R0, ATTR_CTL+4	: 1152
			08	AE	D4 0001B	CLRL	ATTR_CTL+8	: 1153
04	AO	04	AO	10	9C 0001B	ROTL	#16, 4(R0), 4(R0)	: 1156
08	AO	08	AO	10	9C 00021	ROTL	#16, 8(R0), 8(R0)	: 1157
				7E	D4 00027 1\$:	CLRL	-(SP)	: 1164
			04	AE	9F 00029	PUSHAB	ATTR_CTL	
				7E	7C 0002C	CLRQ	-(SP)	
				7E	7C 0002E	CLRQ	-(SP)	
				7E	7C 00030	CLRQ	-(SP)	
			30	AE	9F 00032	PUSHAB	IO STATUS	
				34	DD 00035	PUSHL	#52	
			04	AC	DD 00037	PUSHL	CHANNEL	
				1A	DD 0003A	PUSHL	#26	
		00000000G	00	0C	FB 0003C	CALLS	#12, COMMON_IO	
			07	50	E9 00043	BLBC	STATUS, 2\$: 1165
			50	AE	3C 00046	MOVZWL	IO STATUS, STATUS	
			09	50	E8 0004A	BLBS	STATUS, 3\$: 1166
				50	DD 0004D 2\$:	PUSHL	STATUS	
		00000000G	00	01	FB 0004F	CALLS	#1, LIB\$STOP	
				04	00056 3\$:	RET		: 1168

; Routine Size: 87 bytes, Routine Base: \$CODE\$ + 03DF

```

: 644 1169 1 ROUTINE READ_VIRTUAL (CHANNEL, VBN) : NOVALUE =
: 645 1170 1
: 646 1171 1 ++
: 647 1172 1
: 648 1173 1 FUNCTIONAL DESCRIPTION:
: 649 1174 1
: 650 1175 1 This routine reads the indicated virtual block in the given channel.
: 651 1176 1
: 652 1177 1
: 653 1178 1 CALLING SEQUENCE:
: 654 1179 1 READ_VIRTUAL (ARG1, ARG2)
: 655 1180 1
: 656 1181 1 INPUT PARAMETERS:
: 657 1182 1 ARG1: channel number to use
: 658 1183 1 ARG2: VBN to read
: 659 1184 1
: 660 1185 1 IMPLICIT INPUTS:
: 661 1186 1 NONE
: 662 1187 1
: 663 1188 1 OUTPUT PARAMETERS:
: 664 1189 1 NONE
: 665 1190 1
: 666 1191 1 IMPLICIT OUTPUTS:
: 667 1192 1 BUFFER: contains block read
: 668 1193 1
: 669 1194 1 ROUTINE VALUE:
: 670 1195 1 NONE
: 671 1196 1
: 672 1197 1 SIDE EFFECTS:
: 673 1198 1 NONE
: 674 1199 1
: 675 1200 1 --
: 676 1201 1
: 677 1202 2 BEGIN
: 678 1203 2
: 679 1204 2
: 680 1205 2 LOCAL
: 681 1206 2 STATUS, ! status return
: 682 1207 2 IO_STATUS : VECTOR [4, WORD]; ! I/O status block
: 683 1208 2
: 684 P 1209 2 STATUS = DO_IO (CHAN = .CHANNEL,
: 685 P 1210 2 FUNC = IOS_READVBLK,
: 686 P 1211 2 IOSB = IO_STATUS,
: 687 P 1212 2 P1 = BUFFER,
: 688 P 1213 2 P2 = 512,
: 689 P 1214 2 P3 = .VBN
: 690 1215 2 );
: 691 1216 2 IF .STATUS THEN STATUS = .IO_STATUS[0];
: 692 1217 2 IF NOT .STATUS THEN ERR_EXIT(.STATUS);
: 693 1218 2
: 694 1219 1 END; ! end of routine READ_VIRTUAL

```

0000 00000 READ_VIRTUAL:

BINDVL
V04-000

M 1
16-Sep-1984 01:10:14
14-Sep-1984 12:45:16

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[MOUNT.SRC]BINDVL.B32;1
Page 23
(7)

5E		08	C2	00002	.WORD	Save nothing	:	1169
		7E	7C	00005	SUBL2	#8, SP	:	
		7E	D4	00007	CLRQ	-(SP)	:	1215
					CLRL	-(SP)	:	
		08	AC	DD	PUSHL	VBN	:	
	7E	0200	8F	3C	MOVZWL	#512, -(SP)	:	
		0000G	CF	9F	PUSHAB	BUFFER	:	
			7E	7C	CLRQ	-(SP)	:	
		20	AE	9F	PUSHAB	IO STATUS	:	
			31	DD	PUSHL	#49	:	
		04	AC	DD	PUSHL	CHANNEL	:	
			1A	DD	PUSHL	#26	:	
00000000G	00		0C	FB	CALLS	#12, COMMON_IO	:	
	06		50	E9	BLBC	STATUS, 1\$:	1216
	50		6E	3C	MOVZWL	IO STATUS, STATUS	:	
	09		50	E8	BLBS	STATUS, 2\$:	1217
			50	DD	PUSHL	STATUS	:	
00000000G	00		01	FB	CALLS	#1, LIB\$STOP	:	
			04	0003A	RET		:	1219

; Routine Size: 59 bytes, Routine Base: \$CODE\$ + 0436

CHI
VO

```

696 1220 1 ROUTINE WRITE_VIRTUAL (CHANNEL, VBN) : NOVALUE =
697 1221 1
698 1222 1 !++
699 1223 1
700 1224 1 FUNCTIONAL DESCRIPTION:
701 1225 1
702 1226 1 This routine writes the indicated virtual block in the given channel.
703 1227 1
704 1228 1
705 1229 1 CALLING SEQUENCE:
706 1230 1 WRITE_VIRTUAL (ARG1, ARG2)
707 1231 1
708 1232 1 INPUT PARAMETERS:
709 1233 1 ARG1: channel number to use
710 1234 1 ARG2: VBN to write
711 1235 1
712 1236 1 IMPLICIT INPUTS:
713 1237 1 BUFFER: contains block to be written
714 1238 1
715 1239 1 OUTPUT PARAMETERS:
716 1240 1 NONE
717 1241 1
718 1242 1 IMPLICIT OUTPUTS:
719 1243 1 NONE
720 1244 1
721 1245 1 ROUTINE VALUE:
722 1246 1 NONE
723 1247 1
724 1248 1 SIDE EFFECTS:
725 1249 1 NONE
726 1250 1
727 1251 1 --
728 1252 1
729 1253 2 BEGIN
730 1254 2
731 1255 2
732 1256 2 LOCAL
733 1257 2 STATUS, ! status return
734 1258 2 IO_STATUS : VECTOR [4, WORD]; ! I/O status block
735 1259 2
736 P 1260 2 STATUS = DO_IO (CHAN = .CHANNEL,
737 P 1261 2 FUNC = IOS$WRITEVBLK,
738 P 1262 2 IOSB = IO_STATUS,
739 P 1263 2 P1 = BUFFER,
740 P 1264 2 P2 = 512,
741 P 1265 2 P3 = .VBN
742 1266 2 );
743 1267 2 IF .STATUS THEN STATUS = .IO_STATUS[0];
744 1268 2 IF NOT .STATUS THEN ERR_EXIT(.STATUS);
745 1269 2
746 1270 1 END; ! end of routine WRITE_VIRTUAL

```

0000 00000 WRITE_VIRTUAL:

BINDVL
V04-000

B 2
16-Sep-1984 01:10:14
14-Sep-1984 12:45:16

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[MOUNT.SRC]BINDVL.B32;1
Page 25
(8)

5E		08	C2	00002	.WORD	Save nothing	:	1220
		7E	7C	00005	SUBL2	#8, SP	:	
		7E	D4	00007	CLRQ	-(SP)	:	1266
					CLRL	-(SP)	:	
		08	AC	DD 00009	PUSHL	VBN	:	
7E	0200	8F	3C	0000C	MOVZWL	#512, -(SP)	:	
	0000G	C7	9F	00011	PUSHAB	BUFFER	:	
		7E	7C	00015	CLRQ	-(SP)	:	
		20	AE	9F 00017	PUSHAB	IO STATUS	:	
			30	DD 0001A	PUSHL	#48	:	
		04	AC	DD 0001C	PUSHL	CHANNEL	:	
			1A	DD 0001F	PUSHL	#26	:	
00000000G	00	0C	FB	00021	CALLS	#12, COMMON_IO	:	
	06	50	E9	00028	BLBC	STATUS, 1\$:	1267
	50	6E	3C	0002B	MOVZWL	IO STATUS, STATUS	:	
	09	50	E8	0002E	BLBS	STATUS, 2\$:	1268
		50	DD	00031 1\$:	PUSHL	STATUS	:	
00000000G	00	01	FB	00033	CALLS	#1, LIB\$STOP	:	
		04	0003A 2\$:	RET			:	1270

; Routine Size: 59 bytes, Routine Base: \$CODE\$ + 0471


```

748 1271 1 |
749 1272 1 | The following routine is obsolete. It is commented out for historical
750 1273 1 | reasons only.
751 1274 1 |
752 1275 1 |
753 1276 1 | ROUTINE UNLOCK_INDEXF =
754 1277 1 |
755 1278 1 | ++
756 1279 1 |
757 1280 1 | FUNCTIONAL DESCRIPTION:
758 1281 1 |
759 1282 1 |     This routine zeroes the lock count in the index file FCB's of the
760 1283 1 |     volume being mounted /BIND and of RVN 1. In addition, it resets
761 1284 1 |     the LOG_IO and PHY_IO privilege bits to the process' unamplified
762 1285 1 |     bits, to allow the file system's protection check to work. Once
763 1286 1 |     we have a separate privilege bit to grant SYSTEM file access, this
764 1287 1 |     kluge can be removed. It must be called in kernel mode.
765 1288 1 |
766 1289 1 |
767 1290 1 | CALLING SEQUENCE:
768 1291 1 |     UNLOCK_INDEXF ( )
769 1292 1 |
770 1293 1 | INPUT PARAMETERS:
771 1294 1 |     NONE
772 1295 1 |
773 1296 1 | IMPLICIT INPUTS:
774 1297 1 |     CHANNEL: channel number assigned to current volume
775 1298 1 |
776 1299 1 | OUTPUT PARAMETERS:
777 1300 1 |     NONE
778 1301 1 |
779 1302 1 | IMPLICIT OUTPUTS:
780 1303 1 |     NONE
781 1304 1 |
782 1305 1 | ROUTINE VALUE:
783 1306 1 |     1 if all OK
784 1307 1 |     MOUN$_RVN1NOTMT if RVN 1 is not on line
785 1308 1 |
786 1309 1 | SIDE EFFECTS:
787 1310 1 |     Lock count in index file FCB's zeroed
788 1311 1 |
789 1312 1 | --
790 1313 1 |
791 1314 1 | BEGIN
792 1315 1 |
793 1316 1 | LOCAL
794 1317 1 |     PRIVILEGE_MASK : REF BBLOCK, ! pointer to current privilege mask
795 1318 1 |     UCB             : REF BBLOCK, ! address of UCB
796 1319 1 |     UCB1            : REF BBLOCK, ! address of UCB of RVN 1
797 1320 1 |     FCB             : REF BBLOCK, ! address of index file FCB
798 1321 1 |     RVT             : REF BBLOCK, ! address of relative volume table
799 1322 1 |
800 1323 1 | EXTERNAL
801 1324 1 |     CLEANUP_FLAGS  : BITVECTOR, ! cleanup action flags
802 1325 1 |     CTL$GL_PHD     : REF BBLOCK ADDRESSING_MODE (ABSOLUTE),
803 1326 1 |                     ! pointer to process header
804 1327 1 |     CTL$GQ_PROCPRIV : BBLOCK ADDRESSING_MODE (ABSOLUTE);

```



```

: 805      1328 1 |
: 806      1329 1 |
: 807      1330 1 |
: 808      1331 1 | EXTERNAL ROUTINE
: 809      1332 1 |     GET_CHANNELUCB;
: 810      1333 1 |
: 811      1334 1 |
: 812      1335 1 | Chase through the I/O database to find the index file FCB's. The present lock
: 813      1336 1 | count is saved so it can be restored later.
: 814      1337 1 |
: 815      1338 1 | UCB = GET_CHANNELUCB (.CHANNEL);
: 816      1339 1 | RVT = .BBLOCK [.UCB[UCB$L_VCB], VCB$L_RVT];
: 817      1340 1 | UCB1 = .RVT[RVT$L_UCBLST];
: 818      1341 1 | IF .UCB1 EQL 0 THEN RETURN MOUN$_RVN1NOTMT;
: 819      1342 1 |
: 820      1343 1 | SET_IPL (IPL$_SYNCH);
: 821      1344 1 | FCB = .BBLOCK [.UCB1[UCB$L_VCB], VCB$L_FCBFL];
: 822      1345 1 | LOCK_COUNT1 = .FCB[FCB$L_CNT];
: 823      1346 1 | FCB[FCB$L_CNT] = 0;
: 824      1347 1 |
: 825      1348 1 | FCB = .BBLOCK [.UCB[UCB$L_VCB], VCB$L_FCBFL];
: 826      1349 1 | LOCK_COUNT = .FCB[FCB$L_CNT];
: 827      1350 1 | FCB[FCB$L_CNT] = 0;
: 828      1351 1 | SET_IPL (0);
: 829      1352 1 |
: 830      1353 1 | Reduce LOG_IO and PHY_IO privileges to the normal process values.
: 831      1354 1 |
: 832      1355 1 |
: 833      1356 1 | PRIVILEGE_MASK = CTL$GL_PHD[PHD$Q_PRIVMSK];
: 834      1357 1 | IF NOT .CTL$GQ_PROCPRIV[PRV$_LOG_IO] THEN PRIVILEGE_MASK[PRV$_LOG_IO] = 0;
: 835      1358 1 | IF NOT .CTL$GQ_PROCPRIV[PRV$_PHY_IO] THEN PRIVILEGE_MASK[PRV$_PHY_IO] = 0;
: 836      1359 1 |
: 837      1360 1 | CLEANUP_FLAGS[CLF_RELOCK] = 1;
: 838      1361 1 |
: 839      1362 1 | RETURN 1;
: 840      1363 1 |
: 841      1364 1 | END;

```

! end of routine UNLOCK_INDEXF


```

843 1365 1 |
844 1366 1 | The following routine is obsolete. It is commented out for historical
845 1367 1 | reasons only.
846 1368 1 |
847 1369 1 | ROUTINE LOCK_INDEXF =
848 1370 1 |
849 1371 1 | ++
850 1372 1 |
851 1373 1 | FUNCTIONAL DESCRIPTION:
852 1374 1 |
853 1375 1 | This routine restores the lock counts to the index file FCB's of
854 1376 1 | the volume being mounted and of RVN 1. It also set the LOG_IO and
855 1377 1 | PHY_IO bits in the process privilege mask. It must be called in
856 1378 1 | kernel mode.
857 1379 1 |
858 1380 1 |
859 1381 1 | CALLING SEQUENCE:
860 1382 1 | LOCK_INDEXF ( )
861 1383 1 |
862 1384 1 | INPUT PARAMETERS:
863 1385 1 | NONE
864 1386 1 |
865 1387 1 | IMPLICIT INPUTS:
866 1388 1 | CHANNEL: channel assigned to volume
867 1389 1 |
868 1390 1 | OUTPUT PARAMETERS:
869 1391 1 | NONE
870 1392 1 |
871 1393 1 | IMPLICIT OUTPUTS:
872 1394 1 | NONE
873 1395 1 |
874 1396 1 | ROUTINE VALUE:
875 1397 1 | 1
876 1398 1 |
877 1399 1 | SIDE EFFECTS:
878 1400 1 | Lock counts in FCB's restored
879 1401 1 |
880 1402 1 | --
881 1403 1 |
882 1404 1 | BEGIN
883 1405 1 |
884 1406 1 | LOCAL
885 1407 1 | PRIVILEGE_MASK : REF BBLOCK, ! pointer to current privilege mask
886 1408 1 | UCB : REF BBLOCK, ! address of UCB
887 1409 1 | UCB1 : REF BBLOCK, ! address of UCB of RVN 1
888 1410 1 | FCB : REF BBLOCK, ! address of index file FCB
889 1411 1 | RVT : REF BBLOCK; ! address of relative volume table
890 1412 1 |
891 1413 1 | EXTERNAL
892 1414 1 | CLEANUP_FLAGS : BITVECTOR, ! cleanup action flags
893 1415 1 | CTL$GL_PHD : REF BBLOCK ADDRESSING_MODE (ABSOLUTE);
894 1416 1 | ! pointer to process header
895 1417 1 |
896 1418 1 | EXTERNAL ROUTINE
897 1419 1 | GET_CHANNELUCB; ! get UCB address of channel
898 1420 1 |
899 1421 1 |

```



```

: 900      1422 1 : Find the FCB's in the I/O database and add in the old counts (thus allowing
: 901      1423 1 : for changes in the meantime).
: 902      1424 1 :
: 903      1425 1 :
: 904      1426 1 : UCB = GET_CHANNELUCB (.CHANNEL);
: 905      1427 1 : FCB = .BBLOCK [.UCB[UCB$L_VCB], VCB$L_FCBFL];
: 906      1428 1 : FCB[FCB$W_LCNT] = .FCB[FCB$W_LCNT] + .LOCK_COUNT;
: 907      1429 1 :
: 908      1430 1 : RVT = .BBLOCK [.UCB[UCB$L_VCB], VCB$L_RVT];
: 909      1431 1 : UCB1 = .RVT[RVT$L_UCBLST];
: 910      1432 1 : IF .UCB1 NEQ 0 AND .UCB1 NEQ .UCB
: 911      1433 1 : THEN
: 912      1434 1 :     BEGIN
: 913      1435 1 :         FCB = .BBLOCK [.UCB1[UCB$L_VCB], VCB$L_FCBFL];
: 914      1436 1 :         FCB[FCB$W_LCNT] = .FCB[FCB$W_LCNT] + .LOCK_COUNT1;
: 915      1437 1 :     END;
: 916      1438 1 :
: 917      1439 1 : Turn on the LOG_IO and PHY_IO privilege bits again.
: 918      1440 1 :
: 919      1441 1 :
: 920      1442 1 : PRIVILEGE_MASK = CTL$GL_PHD[PHD$Q_PRIVMSK];
: 921      1443 1 : PRIVILEGE_MASK[PRV$V_LOG_IO] = 1;
: 922      1444 1 : PRIVILEGE_MASK[PRV$V_PHY_IO] = 1;
: 923      1445 1 :
: 924      1446 1 : CLEANUP_FLAGS[CLF_RELOCK] = 0;
: 925      1447 1 : RETURN T;
: 926      1448 1 :
: 927      1449 1 : END;

```

! end of routine LOCK_INDEXF

```

929 1450 1 ROUTINE BIND_HANDLER (SIGNAL, MECHANISM) =
930 1451 1
931 1452 1 ++
932 1453 1
933 1454 1 FUNCTIONAL DESCRIPTION:
934 1455 1
935 1456 1 This routine is the condition handler for the /BIND processing.
936 1457 1 It closes files and deassigns channels as necessary, and resignals
937 1458 1 the error to the main condition handler.
938 1459 1
939 1460 1
940 1461 1 CALLING SEQUENCE:
941 1462 1 BIND_HANDLER (ARG1, ARG2)
942 1463 1
943 1464 1 INPUT PARAMETERS:
944 1465 1 ARG1: address of signal array
945 1466 1 ARG2: address of mechanism array
946 1467 1
947 1468 1 IMPLICIT INPUTS:
948 1469 1 NONE
949 1470 1
950 1471 1 OUTPUT PARAMETERS:
951 1472 1 NONE
952 1473 1
953 1474 1 IMPLICIT OUTPUTS:
954 1475 1 NONE
955 1476 1
956 1477 1 ROUTINE VALUE:
957 1478 1 NONE
958 1479 1
959 1480 1 SIDE EFFECTS:
960 1481 1 various cleanups
961 1482 1
962 1483 1 --
963 1484 1
964 1485 2 BEGIN
965 1486 2
966 1487 2 MAP
967 1488 2 SIGNAL : REF BBLOCK, ! signal array
968 1489 2 MECHANISM : REF BBLOCK; ! mechanism array
969 1490 2
970 1491 2 EXTERNAL
971 1492 2 CLEANUP_FLAGS : BITVECTOR; ! cleanup status flags
972 1493 2
973 1494 2
974 1495 2 ! Do cleanup as indicated by the status flags. Close files and deassign
975 1496 2 ! channels.
976 1497 2
977 1498 2
978 1499 2 IF .BBLOCK [SIGNAL[CHFSL_SIG_NAME], ST$V_SEVERITY] EQL ST$K_SEVERE
979 1500 2 THEN
980 1501 2 BEGIN
981 1502 2
982 1503 2 IF .CHANNEL2 NEQ 0
983 1504 2 THEN
984 1505 2 BEGIN
985 1506 2 DO_IO (CHAN = .CHANNEL2, FUNC = IO$_DEACCESS);

```


BINDVL
V04-000

H 2
16-Sep-1984 01:10:14
14-Sep-1984 12:45:16

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[MOUNT.SRC]BINDVL.B32;1 (11)

Page 31

```

: 986      1507 4      $DASSGN (CHAN = .CHANNEL2);
: 987      1508 4      CHANNEL2 = 0;
: 988      1509 3      END;
: 989      1510 3
: 990      1511 3      IF .CHANNEL3 NEQ 0
: 991      1512 3      THEN
: 992      1513 4      BEGIN
: 993      1514 4      DO IO (CHAN = .CHANNEL3, FUNC = IO$_DEACCESS);
: 994      1515 4      $DASSGN (CHAN = .CHANNEL3);
: 995      1516 4      CHANNEL3 = 0;
: 996      1517 3      END;
: 997      1518 3
: 998      1519 3
: 999      1520 3      IF .CLEANUP_FLAGS[CLF_RELOCK]
1000      1521 3      THEN KERNEL_CALL (LOCK_INDEXF);
: 1001      1522 3
: 1002      1523 3
: 1003      1524 2      END;
: 1004      1525 2
: 1005      1526 2      RETURN SSS_RESIGNAL;
: 1006      1527 2
: 1007      1528 1      END;
                                ! end of routine BIND_HANDLER
```

.EXTRN CLEANUP_FLAGS

				001C 00000 BIND_HANDLER:				
				54 00000000G	00 9E 00002	.WORD	Save R2,R3,R4	1450
				53 00000000G	00 9E 00009	MOVAB	SYSSDASSGN, R4	
				52 0000' 04	CF 9E 00010	MOVAB	COMMON_IO, R3	
04	04	A0		50 04	AC D0 00015	MOVAB	CHANNEL2, R2	1499
				03	00 ED 00019	MOVL	SIGNAL, R0	
					41 12 0001F	CMPZV	#0, #3, 4(R0), #4	
				50	62 3C 00021	BNEQ	2\$	1503
					1A 13 00024	MOVZWL	CHANNEL2, R0	
					7E 7C 00026	BEQL	1\$	1506
					7E 7C 00028	CLRQ	-(SP)	
					7E 7C 0002A	CLRQ	-(SP)	
					7E 7C 0002C	CLRQ	-(SP)	
				7E	34 7D 0002E	CLRQ	-(SP)	
					50 DD 00031	MOVQ	#52, -(SP)	
					1A DD 00033	PUSHL	R0	
				63	0C FB 00035	PUSHL	#26	
				7E	62 3C 00038	CALLS	#12, COMMON_IO	1507
				64	01 FB 0003B	MOVZWL	CHANNEL2, -(SP)	
					62 B4 0003E	CALLS	#1, SYSSDASSGN	1508
				50	A2 3C 00040 1\$:	CLRQ	CHANNEL2	1511
					1C 13 00044	MOVZWL	CHANNEL3, R0	
					7E 7C 00046	BEQL	2\$	
					7E 7C 00048	CLRQ	-(SP)	1514
					7E 7C 0004A	CLRQ	-(SP)	
					7E 7C 0004C	CLRQ	-(SP)	
				7E	34 7D 0004E	CLRQ	-(SP)	
					50 DD 00051	MOVQ	#52, -(SP)	
					1A DD 00053	PUSHL	R0	
						PUSHL	#26	

BINDVL
V04-000

1 2
16-Sep-1984 01:10:14
14-Sep-1984 12:45:16

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[MOUNT.SRC]BINDVL.B32;1 (11)

Page 32

63		0C	FB	00055	CALLS	#12, COMMON IO	:	
7E	02	A2	3C	00058	MOVZWL	CHANNEL3, -(SP)	:	1515
64		01	FB	0005C	CALLS	#1, SYS\$DASSGN	:	
	02	A2	B4	0005F	CLRW	CHANNEL3	:	1516
50	0918	8F	3C	00062	MOVZWL	#2328, R0	:	1526
			04	00067	RET		:	1528

: Routine Size: 104 bytes, Routine Base: \$CODE\$ + 04AC

: 1008 1529 1
: 1009 1530 1 END
: 1010 1531 0 ELUDOM

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	12	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	28	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODE\$	1300	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	64	0	1000	00:01.9

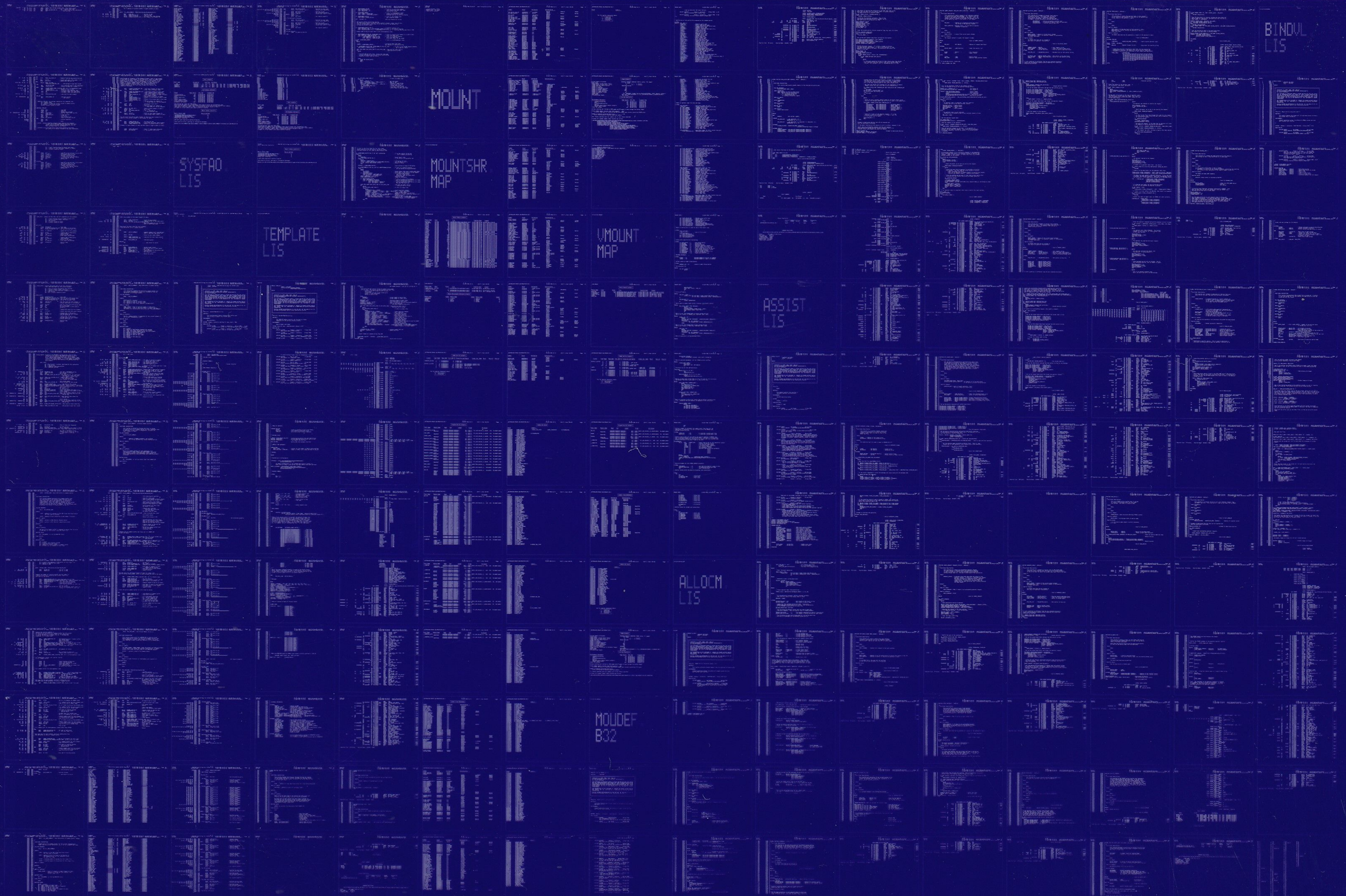
COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:BINDVL/OBJ=OBJ\$:BINDVL MSRC\$:BINDVL/UPDATE=(ENH\$:BINDVL)

: Size: 1300 code + 40 data bytes
: Run Time: 00:30.4
: Elapsed Time: 01:11.7
: Lines/CPU Min: 3021
: Lexemes/CPU-Min: 25229
: Memory Used: 203 pages
: Compilation Complete

0243 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0244

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY